International Journal of Security and Its Applications
Vol.8, No.6 (2014), pp.197-210
http://dx.doi.org/10.14257/ijsia.2014.8.6.18

Batch Auditing for Multi-client Dynamic Data in Multi-cloud Storage

Xin Liu and Yujia Jiang

[1]*College of Architecture and Artistic Design, Hunan Institute of Technology, Hengyang, 421001, China*
liuxin7890@163.com, jiangyujia7890@163.com

Abstract

Cloud storage introduces many challenges due to the security and integrity threats toward user's outsourced data. Many auditing protocols have been proposed, but the majority of them could only serve in single cloud environment. This paper proposes an efficient auditing protocol, which supports batch auditing for multiple data files in multi-cloud environment. By utilizing the bilinear map, the proposed protocol achieves full stateless and transparent verification. By constructing a Merkle Hash Tree, the proposed protocol can resist the replace attack and support dynamic operation of data. In addition, our protocol protects the position information of the data blocks by generating fake data blocks to confuse the organizer. The performance analysis demonstrates the efficiency of the protocol.

Keywords: *Multi-cloud, Batch auditing, BLS signature, Homomorphic authenticators*

1. Introduction

Cloud storage service is an important service of cloud computing. The data owners can access their data via networks at any time and from anywhere by upload data into the cloud. However, many users are hesitant to use such a novel paradigm because of the security and integrity threats toward their outsourced data. On one hand, the loss of data may happen in a infrastructure. One the other hand, the CSPs could be dishonest to hide data loss accidents to maintain the reputation. Therefore, it is needed for the data owners to check the integrity of the cloud data. Many provable data possession (PDP) protocols have been proposed to verify the integrity of the cloud data [1-9]. Ateniese, *et al.,* [1] defined PDP model with public verification. The authors applied homomorphic linear authenticators to verify the blocks of the file. Erway, *et al.,* [3] presented a skip list-based dynamics PDP model which supports dynamic operation. The scheme supports both the public stateless verification and fully dynamic data update. Wang, *et al.,* [6] proposed a scheme supporting privacy-preserving public auditing, which was also extended to enable batch auditing.

The PDP schemes above mainly address integrity auditing issues for a single CSP. In practice, users would like to store the data on multi-cloud with a distributed manner to reduce the threats of data integrity [10]. In this context, multi-cloud is made up of multiple clouds, and each CSP has a different level of quality of service. Within multi-cloud, an organization can offer and manage in-house and out-house resources [11-13]. For multi-cloud storage scenario, Zhu, *et al.,* [11, 12] designed a cooperative provable data possession protocol based on the techniques of fragment structure, hash index hierarchy, and homomorphic verifiable response. In Zhu, *et al.,* scheme, one of CSPs takes the responsibility as the organizer to cooperate the communication between CSPs and verifier. The scheme supports the efficient verification of data stored on multi-cloud servers, but is not a stateless verification protocol

ISSN: 1738-9976 IJSIA

because the verifier needs to store and update verification information. In addition, it is proved by Wang and Zhang [9] that, the malicious CSPs or organizer in this scheme even be able to pass validation without storing users' data. Etemad, *et al.*, [14] proposed a distributed and replicated PDP scheme which could enable CSPs to hide the internal structure from the client. This scheme is only designed to audit data files one by one. With the wide adoption of cloud computing, the auditor may concurrently receive multiple auditing delegations from different users. If the multiple tasks can be handled in a batch manner, the protocol efficiency will be greatly improved. Yang, *et al.*, [15] proposed a privacy-preserving auditing protocol which could deal with the verification tasks from different users in a batch manner in multi-cloud scenario. However, all of the PDP protocols for multi-cloud scenario above [9, 11, 12, 14, 15] cannot be regarded as the full stateless and transparent verification, because these protocols need to store the position information of the data blocks on the organizer or TPA.

In this paper, we develop an efficient auditing mechanism, which supports batch auditing for multiple data files in multi-cloud environment. By utilizing the bilinear map, the proposed protocol can aggregate the verification task from different users to reduce the computing overhead of the auditor. By constructing a sequence-enforced Merkle Hash Tree, the proposed protocol can resist the replace attack. In addition, our protocol protects the position information of the data blocks by generating fake data blocks to confuse the organizer, so as to achieve full stateless and transparent verification.

The rest of the paper is organized as follows. Section 2 presents the system model, threat model, design goals, and the preliminaries. Section 3 describes the proposed protocol in detail. We provide security analysis and performance evaluation in Section 4. Section 5 concludes the proposed protocol.

2. Problem Statement

2.1 The System Model

As illustrated in Figure 1, we consider a multi-cloud storage service model which is adopted by some previous works [10, 11]. The model involves three different entities cloud users, multi-cloud and third party auditor.

The cloud users have a number of data files to be stored in multiple clouds. They have the authority to access the stored data.

The multi-cloud consists of multiple Cloud Server Providers (CSPs). They provide data storage service and have enough storage space and significant computation resources. In this paper, to reduce the communication burden of verifier, one of CSPs is designated as an organizer for auditing purpose. For example, the Zoho cloud in Figure 1 is considered as an organizer. In our scheme, the organizer takes the responsibility to distribute the auditing challenge and aggregate the proof from multiple clouds. In this paper, we suppose that the CSPs cannot communicate with each other apart from the organizer for the auditing issues, and also, the verifier can only contact with the organizer.

The Third Party Auditor (TPA) has a more powerful computation and communication ability than regular cloud users. In cloud storage system, none of cloud service providers or users could be guaranteed to provide unbiased auditing result. Thus, third party auditing is a natural choice. Moreover, by resorting to TPA, users can be relieved from the burden of checking the integrity of outsource data.

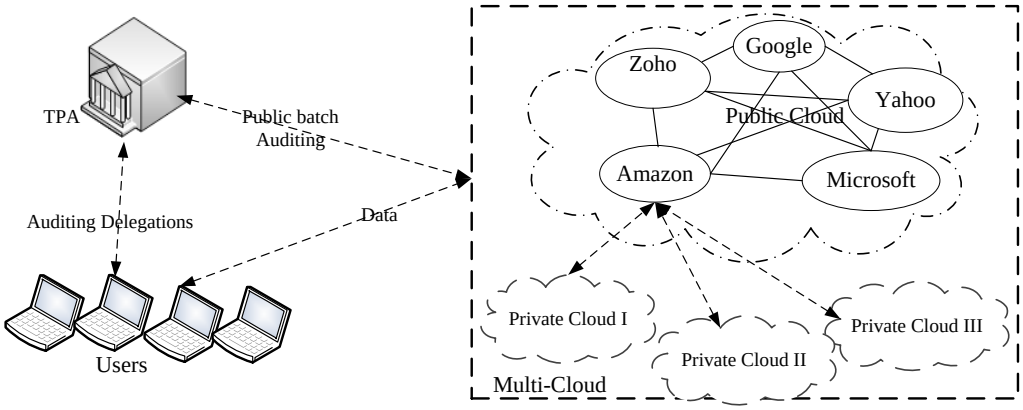

Figure 1. The Frame Work of the Data Storage Auditing

2.2. The Threat Model

First, the TPA is considered as 'honest-but-curious'. It performs honestly the whole auditing protocol but is curious about the received data. Thus, the proposed protocol needs to protect the data privacy against TPA.

Second, the CSPs in the system model could be dishonest. The CSPs are the main data integrity threats toward users' data. For example, when the challenged data block and data tag is corrupted or discarded, they may choose another valid pair of data block and data tag to pass the auditing.

Third, the organizer in the system model is one of the CSPs. Besides the properties of the common CSPs, the organizer may curious about the additional storage information such as the information about in which cloud the specific data blocks are stored. Most of the existing auditing protocols leak this information directly [9, 11, 12, 14, 15], and our protocol tries to protect it by introducing some random data blocks.

In addition, it is supposed that the CSPs, organizer and TPA will not collude with each other during the auditing process. In order to authorize the organizer and CSPs to respond to the verification challenge from TPA, the users issue a certificate on TPA's public key, and all the challenge requests from the TPA are authenticated by such a certificate. The subsequent presentation ignores these authentication handshakes.

2.3. Design Goals

In order to enable secure and high-efficiency over the integrity auditability of outsourced data under the above models, the protocol is designed to achieve the following goals:

- *Security aspect*. The protocol should meet privacy-preserving property. That is to say TPA and the organizer could not learn any knowledge about the outsourced data content and the details of data storage during the efficient auditing process. In addition, the protocol should be able to resist various attach from CSPs.
- *Usability aspect*. Beside the security requirement, the proposed protocol needs to meet some other properties to be practical. 1) *Public batch auditing*: The protocol enables TPA to verify the integrity of multiple data files stored in multi-cloud simultaneously; 2) *Stateless verification*: TPA is not necessary to maintain and update state in proposed protocol. Because statelessness is much-

needed in PDP systems with public verifiability so that anyone can play the role of verifier.

- *Performance aspect*. The protocol should have higher efficiency than single auditing for each user's file, while meeting the above properties.

2.4 Notations and Preliminaries

Sequence-enforced Merkle Hash Tree (SMHT). A Merkle Hash Tree (MHT) [16] is a tree structure designed to authenticate a set of data efficiently. It is usually a binary tree with the hashes of authentic data values as its leaves. Each internal node in a MHT stores a hash value of the combination of its child nodes' value. Let $H(\cdot)$ and $h(\cdot)$ be a secure map-to-point hash function. Here, the function $H(\cdot)$ is used to compute the hash values of the data blocks. In this paper, we treat $\{T_i = H(m_i)\}_{i\in[1,n]}$ as the authentic data values, which is arranged to be a left-to-right sequence. The function $h(\cdot)$ can take more than one argument as $h(x_1, x_2) = h(x_1 \| x_2)$, where the operator $\|$ denotes concatenation. In this paper, we use a sequence-enforced Merkle Hash Tree (SMHT), which is an improved version of MHT structure as illustrated in Figure 2. In our protocol, each node of SMHT carries two auxiliary values: a hash value and an index. The computation of hash value in a SMHT is similar to a common MHT. The index of a SMHT's node indicates the total leaf nodes in its subtree. Specifically, the index of leaf node is set to be1 to inflect itself. Accordingly, if the left child a and right child b of the node r carried auxiliary values (h_a, n_a) and (h_b, n_b) respectively, the index of the node r is $n_r = n_a + n_b$, and the hash of it is computed as $h_r = h(h_a \| h_b \| n_r)$. Figure 2 depicts an example of a simple SMHT which have four data blocks. With the SMHT, we can verify both the value and the position of chosen data block.

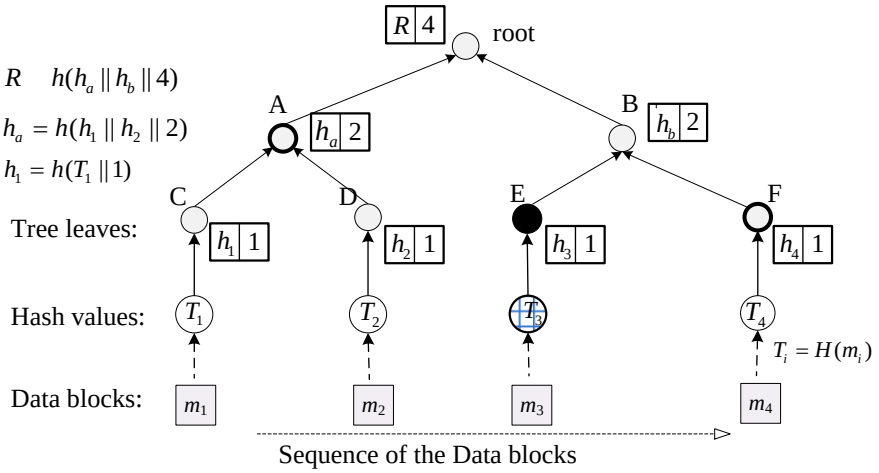

Figure 2. Sequence-enforced Merkle Hash Tree Authentication of Data Blocks

Bilinear Map. Let G_1, G_2 be Gap Diffie-Hellman (GDH) groups of prime order *p*, and let G_T be another multiplicative cyclic group of prime order *p*. Let g_1, g_2 be generators of G_1 and G_2 respectively. For $u \in G_1, v \in G_2$, and all $a, b \in Z_p$, a bilinear map is a

map $e: G_1 \times G_2 \to G_T$ with three properties [5]: 1) Bilinear: $e(u^a, v^b) = e(u,v)^{ab}$;2) Computable: There exists an efficient computable algorithm for computing $e(u,v)$; 3)Non-degenerate: $e(g_1, g_2) \neq 1$.

The key idea of the batch auditing is to use the bilinear map which could aggregate the signatures. For any $k_1, k_2 \in Z_p$, we calculate $u_1 = u^{k_1} \in G_1$, $u_2 = u^{k_2} \in G_1$. Then, for $v \in G_2$, we have

$$e(u_1 u_2, v) = e(u^{k_1+k_2}, v) = e(u,v)^{k_1+k_2} = e(u,v)^{k_1} \cdot e(u,v)^{k_2} = e(u_1, v) \cdot e(u_2, v). \quad (1)$$

3. The Batch Auditing CPDP Protocol

For the sake of clarity, Table 1 lists some signals and descriptions used later in this paper.

Our batch auditing protocol for multi-cloud storage consists of two phases: setup phase and batch audit phase (see Figure 3). During the setup phase, the users preprocess the data files to be stored in the multi-cloud by the algorithms $KeyGen()$ and $TagGen()$. The batch auditing phase is run among CSPs, the organizer and TPA to complete the multiple auditing requests for distinct data files simultaneously.

Table 1. Symbol and Descriptions

Symbol	Descriptions
n	the number of blocks in a file;
s	the number of sectors in each block;
F	the file with $n \times s$ sectors, i.e. $F = \{m_{i,j}\}_{j \in [1,s]}^{i \in [1,n]}$;
θ	the response for the challenge Q;
θ_c	the response from P_c for the challenge Q_c;

The data fragment technique is a data structure that keeps a set of block-tag. With the fragment structure, an outsourced file F is split into n blocks $\{m_1, m_2, ..., m_n\}$, and each block is further split into s sectors $\{m_{i,1}, m_{i,2}, ..., m_{i,s}\}$. Each data block has its physical meanings and can be updated dynamically by the users. We only need to generate one data tag for each data block that consists of s sectors.

In the proposed protocol, we need to use a bilinear map group system, two hash functions and a signature function. Let $S = (p, g, G_1, G_2, G_T, e)$ be a bilinear map group system with generator g , where G_1, G_2 and G_T are multiplicative cyclic groups of prime order p , and $e: G_1 \times G_2 \to G_T$ is a bilinear map. Let $H(\cdot): \{0,1\}^* \to G_1$ be a secure map-to-point hash function, $h(\cdot): G_T \to Z_p$ be another hash function which maps element of G_T to Z_p , and $Sig(\cdot)$ be the signature function. Let $\{U_k\}$ be the set of users, and $\{P_c\}$ be the set of CSPs. The number of CSPs is denoted as C .

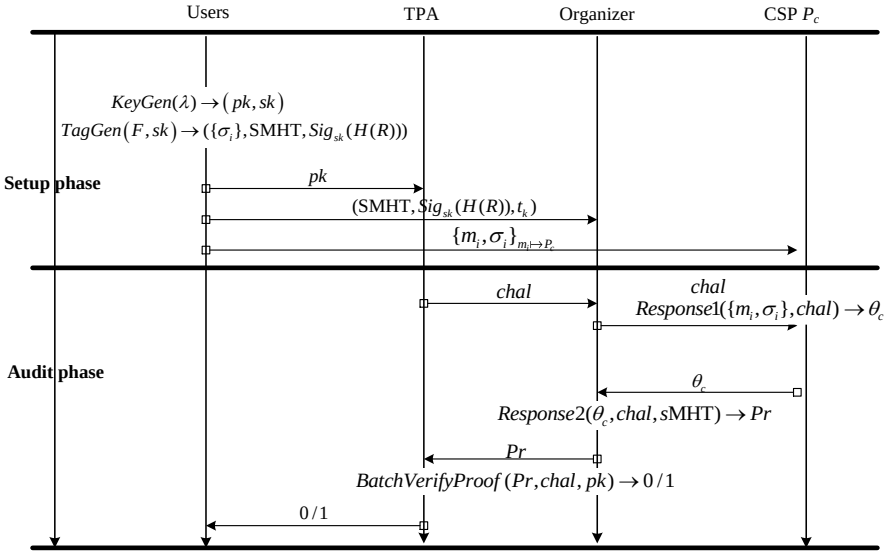

Figure 3. The Framework of Protocol

3.1. Setup Phase

Each user runs setup phase of the protocol as follow:

Step-1: $KeyGen\ (1^{\lambda})$. The k^{th} user U_k generates a signing key pair (ssk_k, spk_k). Select a random $\alpha_k \leftarrow Z_p$, and compute $\beta_k \leftarrow g^{\alpha_k}$. Select s random elements $\{u_{k,1}, u_{k,2}, ..., u_{k,s}\} \subset G_1$. To sum up, the secret key is $sk_k = (\alpha_k, ssk_k)$ and the public parameters are $pk_k = (spk_k, \beta_k, g, \{u_{k,j}\}_{1 \le j \le s})$.

Step-2: $TagGen\ (sk_k, F_k, P)$. The user U_k splits his file F_k into $n_k \times s$ sectors $F_k = \{m_{k,i,j}\}_{\forall i \in [1,n_k], l \in [1,s]} \subset Z_p^{n_k \times s}$, where the i^{th} block of user U_k block $m_{k,i}$ consists of s sectors $\{m_{k,i,1}, m_{k,i,2}, ..., m_{k,i,s}\}$. The user computes $t_k = name_k \| n_k \| Sig_{ssk_k}(name_k \| n_k)$ as the file tag for F_k, where $name_k$ is the file name. Next, the user U_k constructs SMHT with a root R_k, where the leave nodes are an ordered hash values of data blocks $\{T_{k,i} = H(m_{k,i})\}_{i \in [1,n_k]}$ as described in part 2.3. The user also signs R_k with the private key α_k as:

$$Sig_{\alpha_k}(H(R_k)) = (H(R_k))^{\alpha_k}. \tag{2}$$

For each data block $m_{k,i} (\forall i \in [1, n_k])$, the user U_k computes a data tag $\sigma_{k,i}$ as:

$$\sigma_{k,i} = (T_{k,i} \cdot \prod_{j=1}^{s} u_{k,j}^{m_{k,i,j}})^{\alpha_k}. \tag{3}$$

Besides, the user U_k generates an additional random data block denoted as $m_{k,n_k'}$, which is also divided into *s* sectors $\{m_{k,n_k',j}\}_{j \in [1,s]}$. Then the corresponding tag is computed as

$$\sigma_{k,n'} = (H(R_k)^{\frac{1}{C}} \cdot \prod_{j=1}^{s} u_{k,j}^{m_{k,n_k',j}})^{\alpha_k}. \tag{4}$$

After all the parameters have been prepared, the user U_k distributes the data block and tag pairs $(m_{k,i}, \sigma_{k,i})$ to the corresponding cloud service providers, and sends the random data block

and its corresponding tag $(m_{k,n_k}, \sigma_{k,n_k})$ to each cloud service provider. In addition, the user U_k sends the parameters $\{SMHT, Sig_{\alpha_k}(H(R_k)), t_k\}$ to the organizer, and sends the public parameters pk_k to TPA. After data transmission, the user asks TPA to conduct the confirmation auditing to make sure that their data is correctly stored on all the servers. Once confirmed, the user can choose to delete the local copy of the data blocks apart from the secret key. By now, TPA could run the sampling auditing periodically to check the data integrity for users.

3.2. Batch Audit Phase

Suppose that TPA process K auditing sessions of K distinct data files simultaneously. The data files are stored on C CSPs $(\{P_c\} c \in C)$. The audit phase is executed as follows:

Step-1: $GenProof(\{m_i, \sigma_i\}, chal)$ .It is an interactive 5-move protocol among CSPs, an organizer(*O*), and an Auditor (TPA). This process is described in the following.

1) **Retrieve** (*O*→TPA): After receiving the verification request, the organizer sends the file tags $\{t_k\}_{k\in[1,K]}$ to the TPA. The TPA recovers $\{name_k\}_{k\in[1,K]}$ and $\{n_k\}_{k\in[1,K]}$ by using public keys $\{sspk_k\}_{k\in[1,K]}$, and verifies all the signatures. The verification quit by emitting *FALSE* if the file name verification fails.

2) **Challenge1** (*O*←TPA): If the file name is successful verified, the TPA generates challenge index-coefficient message $Q_k = \{(i, v_i)\}_{i\in I_k}$ for $\forall k \in [1, K]$, where $I_k \subseteq [1, n_k]$ specifies the sampled blocks that will be verified, and $v_i \in Z_p$ is random element. TPA chooses a random element $\gamma \leftarrow Z_p$ and computes the set of challenge stamps $\{H_k = \beta_k{}^{\gamma}\}_{k\in[1,K]}$. To sum up, TPA generates the challenge as follow, and sent it to the organizer.

$$chal = (\{Q_k\}_{k\in[1,K]}, \{H_k = \beta_k{}^{\gamma}\}_{k\in[1,K]}). \tag{5}$$

3) **Challenge2** (*P*←*O*): Upon receiving the challenge $chal$ from the TPA, the organizer forwards $chal$ to each P_c .

4) **Response1** (*P*→*O*): For each Q_k , each P_c picks out $Q_{c,k} = \{(i, v_i)\}_{m_{k,i}\mapsto P_c} \subseteq Q_k$. Here, the symbol $m_{k,i} \mapsto P_c$ means that the data block $m_{k,i}$ is stored on P_c .Then, P_c calculates the linear combination of specified data blocks for each file as

$$\mu_{c,k,j} = m_{k,n_k',j} + \sum_{(i,v_i)\in Q_{c,k}} v_i m_{k,i,j} \in Z_p, \tag{6}$$

and generates data proof DP_c and tag proof TP_c as

$$\begin{cases} DP_c = \prod_{j=1}^{s}\prod_{k=1}^{K} e(u_{k,j}, H_k)^{\mu_{c,k,j}} \\ TP_c = \prod_{k=1}^{K}\left(\sigma_{k,n_k'} \cdot \prod_{(i,v_i)\in Q_{c,k}} (\sigma_{k,i,c})^{v_i}\right) \in G_1 \end{cases}. \tag{7}$$

Finally, P_c returns $\theta_c = (DP_c, TP_c)$ to the organizer.

5) **Response2** (*O*→TPA): Upon receiving all the proofs from all the CSPs, the organizer aggregates all of the proof θ_c into a response $\theta = (DP, TP)$, which are calculated as

$$DP = \prod DP_c, \qquad TP = \prod TP_c. \tag{8}$$

In addition, the organizer also provides the TPA with AAI $\{\Omega_{k,i}\}_{i\in I_k,k\in[1,K]}$, which are the siblings of the nodes on the path from the leaves $\{T_{k,i}\}_{i\in I_k,k\in[1,K]}$ to the root R_k of the SMHT. Finally, the organizer responds TPA with proof.

$$Pr = \left(\{\Omega_{k,i}\}_{i\in I_k,k\in[1,K]}, \{Sig_{\alpha_k}(H(R_k))\}_{k\in[1,K]}, \theta\right) \quad (9)$$

Step-2: $BatchVerifyProof(pk, chal, Pr)$. After receiving proof Pr from the organizer, TPA starts to verify to proof. First, for each $k \in [1, K]$ and $i \in I_k$, TPA first verifies the position of $\{T_{k,i}\}_{for\ \forall i\in I_k,k\in[1,k]}$ by checking if the equation $LEFT(T_{k,i}) = i - 1$ holds or not. Second, TPA constructs a verification root R_k' by using $\{T_{k,i}, \Omega_{k,i}\}_{i\in I_k}$ for $\forall k \in [1, K]$, and verifies values of R_k' by checking Eq. (10). Third, if the both authentication above succeeds, TPA verifies data integrity by checking Eq. (11).

$$e(\prod_{k=1}^{K} sig_{\alpha_k}(H(R_k)), g) \stackrel{?}{=} \prod_{k=1}^{K} e(H(R_k'), \beta_k) \quad (10)$$

$$e(TP^{\gamma}, g) \stackrel{?}{=} DP \cdot \prod_{k=1}^{K} e(H(R_k') \cdot \prod_{i\in I} T_{k,i}^{v_i}, H_k) \quad (11)$$

The verification Eq. (11) only holds when all the responses are valid. However it will fail when there is even one single invalid response in the batch auditing. The ratio of invalid responses to the valid could be quite small, and yet a standard batch auditor will reject the entire collection. Thus, in order to further sort out these invalid responses in the batch auditing, the paper can utilize a recursive divide-and-conquer approach (binary search), as suggested by Ferrara, *et al.*, [17]. Specifically, if the batch auditing fails, we can simply divide the collection of responses into two halves, and repeat the auditing on halves via Eq. (11).

3.3. Dynamic Data Operation

In cloud storage, the remotely stored data might not only be accessed but also updated frequently by the users. Thus, supporting dynamic data operation is also very important for the practical application of storage outsourcing service. With the introduction of SMHT authenticator construction, the proposed scheme can efficiently handle fully dynamic data block-level operations including data insertion (I), data modification (M), and data deletion (D).

The organizer executes insertion operation (*ExecUpdate*):

1. Insert a new leaf node $(h(T_{k,*}||1),1)$ after leaf node $(h(T_{k,i}),1)$ and add an internal node $(h(h(T_{k,i}) || h(T_{k,*}) || 2), 2)$ in the original SMHT;
2. Renew all information of node on the path from the internal node to the root;
3. Generate the new root *R*' of the updated SMHT.

Figure 4. The Organizer Executes Insertion Operation on SMHT

Data Insertion (I): Data insertion is a relatively complex operation. If the user U_k wants to insert block $m_{k,*}$ after the i^{th} block $m_{k,i}$, he firstly splits $m_{k,*}$ into s sectors $\{m_{k,*,1}, m_{k,*,2}, ..., m_{k,*,s}\}$ and computes $T_{k,*} = H(m_{k,*})$. Then, user U_k will generate the

corresponding tag $\sigma_{k,*}$ by Eq. (3). Finally, the user sends update request "$update = (\mathrm{I}, i, T_{k,*})$" to the organizer and $\{\mathrm{I}, i, m_{k,*}, \sigma_{k,*}\}$ to a chosen cloud service provider P_c, respectively.

Upon receiving the request, P_c will store the new data-tag pair, and the organizer executes the update operation as shown in Fig. 4. At last, the organizer responses the user with a proof for this operation, $P_{update} = (\Omega_{k,i}, T_{k,i}, Sig_{sk}(H(R_k)), R_k')$ where $\Omega_{k,i}$ is the AAI for authentication of $T_{k,i}$ in the old SMHT.

After receiving the proof for insert operation from the organizer, the user U_k will verify update by Algorithm3. If it outputs *TRUE*, the user signs the new root metadata R_k' by Eq. (2) and updates the file tag t_k, which both will be sent to the organizer for update. Finally, the user challenges the organizer to execute the integrity verification protocol, where the set of challenge contains updated block. If the verification succeeds, delete update information from local storage.

An example of inserting block $(h(T_{k,*}||1),1)$ after $(h_2,1)$ is shown in Figure 5, where a new leaf node and an internal node C' are added to the old SMHT.

Algorithm3 *VerifyUpdate*

input: $P_{update} = (\Omega_{k,i}, T_{k,i}, Sig_{sk}(H(R_k)), R_k')$

output: *TRUE/FALSE*

1: The user U_k generate root R_k using $\{\Omega_{k,i}, T_{k,i}\}$;
2: **if** $e(sig_{sk_k}(H(R_k)), g) \neq e(H(R_k), v_k)$ **then** //authenticate root R_k
3: **return** *FALSE*;
4: **else**
5: The user U_k computes the new root R_k'' value using $\{\Omega_{k,i}, T_{k,i}, T_{k,*}\}$;
6: **if** $R_k'' \neq R_k'$ **then** // authenticate new root R_k
7: **return** *FALSE*;
8: **else**
9: **return** *TRUE*;
10: **end if**
11: **end if**

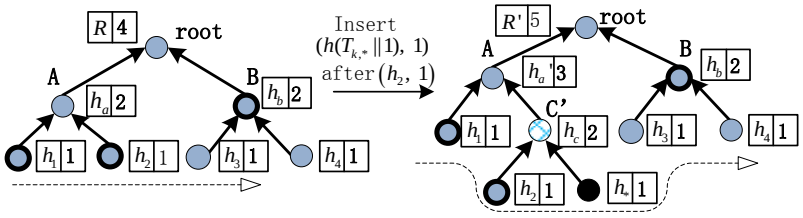

Figure 5. Example of Block Insertion Operation under SMHT

Data Modification (M**)**: Similar to the procedures of the block insertion, upon receiving the *update* request for modifying block, P_c will replace the specified data-tag pair with new one, and the organizer just needs to update all the hashes of nodes from the specified nodes to the root in SMHT, which does not change the tree structure. The detailed process is similar to the block insertion, which is thus omitted here.

Data Deletion (D**)**: The operation of data deletion is only the opposite operation of data insertion. For single block deletion, upon receiving the *update* request for deleting block, P_c will delete the specified data-tag pair from store space, and the organizer will delete the specified leaf node. An example of deleting block $(h_2,1)$ is shown in Figure 6. The details are similar to the procedures of the block insertion.

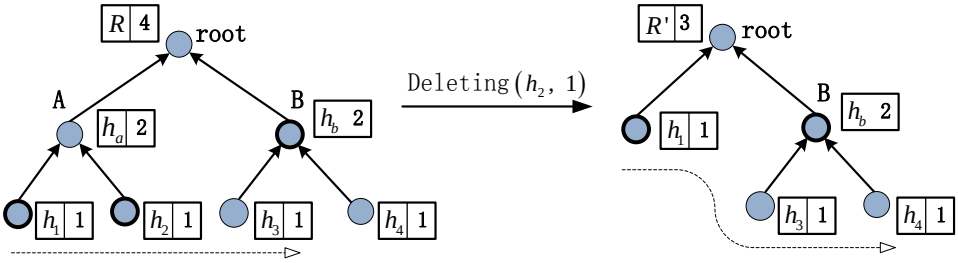

Figure 6. Example of Block Deletion Operation under SMHT

4. Evaluation

4.1. Security Analysis

We evaluate the security of the proposed scheme by analyzing whether it fulfills the goals described in part 2.3, such as privacy-preserving property, public batch auditing and stateless verification.

Theorem 1. TPA and the organizer will not learn any knowledge about the outsourced data.

Proof: First, we show that TPA cannot learn any data privacy from DP . This is because, in Eq. (7), the linear combination of specified data blocks $\mu_{c,k,j}$ is exponent of data block, and TPA cannot compute the value of data block by solving the equations. Besides, with the introduction of the organizer, TPA cannot learn the details of data storage. It is worth noting that, this paper regards the organizer is also untrustworthy. At the setup phase of protocol, user generates a pair of additional data-tag which will be sent to each CSP. At the audit phase, whichever data block is in the challenge request from the TPA, every CSP will send a response to the organizer, which can conceal the details of data storage from the organizer. Thus, the proposed batch auditing protocol for multi-cloud conceals the storage details from both TPA and the organizer.

Theorem 2. TPA can check the integrity of the outsourced data in a stateless and batch manner.

Proof: Based on BLS short signature technology, the scheme introduces the SMHT structure. Before uploading files to the cloud, user will calculate hash value for each file data block, and construct a SMHT structure with these hash values as the leave nodes. Finally, the user computes the signature of root in SMHT, which will be stored to the organizer together with SMHT. So, TPA do not need to keep any information, and he can check the correctness of the data hash values only by verifying the signature of the root, which makes the scheme

achieve the stateless public verification. In fact, the scheme can also resist *Replace attack* of CSPs. Moreover, TPA can check the integrity of multiple data files simultaneously. If the data blocks are not damaged by the cloud server, the equation (11) will be proved to be true. The proof of equation (11) is elaborated as follows:

$$left = e((\prod_{P_c \in P} TP_c)^{\gamma}, g) = e(\prod_{P_c \in P} (\prod_{k=1}^{K} (\sigma_{k,n_k} \cdot \prod_{(i,v_i) \in Q_{c,k}} (\sigma_{k,i,c})^{v_i})), g^{\gamma}) = \prod_{k=1}^{K} e(\prod_{P_c \in P} (\sigma_{k,n_k} \cdot \prod_{(i,v_i) \in Q_{c,k}} (\sigma_{k,i,c})^{v_i}), g^{\gamma})$$

$$= \prod_{k=1}^{K} e(\prod_{P_c \in P} ((H(R_k)^{\frac{1}{C}} \cdot \prod_{j=1}^{s} u_{k,j}^{m_{k,n_k},j})^{\alpha_k} \cdot \prod_{(i,v_i) \in Q_{c,k}} ((T_{k,i} \cdot \prod_{j=1}^{s} u_{k,j}^{m_{k,i,j}})^{\alpha_k})^{v_i}), g^{\gamma})$$

$$= \prod_{k=1}^{K} e(H(R_k) \cdot \prod_{(i,v_i) \in Q_{c,k}} T_{k,i}^{v_i}, H_k) \cdot \prod_{k=1}^{K} e(\prod_{P_c \in P} (\prod_{j=1}^{s} u_{k,j}^{m_{k,n_k},j} \cdot \prod_{(i,v_i) \in Q_{c,k}} (\prod_{j=1}^{s} u_{k,j}^{m_{k,i,j}})^{v_i}), H_k)$$

$$= \prod_{k=1}^{K} e(H(R_k) \cdot \prod_{(i,v_i) \in Q_{c,k}} T_{k,i}^{v_i}, H_k) \cdot \prod_{k=1}^{K} e(\prod_{P_c \in P} \prod_{j=1}^{s} u_{k,j}^{m_{k,n_k},j + \sum_{(i,v_i) \in Q_{c,k}} v_i \cdot m_{k,i,j}}, H_k)$$

$$= \prod_{k=1}^{K} e(H(R_k) \cdot \prod_{(i,v_i) \in Q_{c,k}} T_{k,i}^{v_i}, H_k) \cdot \prod_{P_c \in P} \prod_{j=1}^{s} \prod_{k=1}^{K} e(u_{k,j}, H_k)^{\mu_{c,k,j}}$$

$$= \prod_{k=1}^{K} e(H(R_k) \cdot \prod_{(i,v_i) \in Q_{c,k}} T_{k,i}^{v_i}, H_k) \cdot DP = right$$

4.2. Performance Analysis

Computation Cost: The computation cost of simple algebraic operations and modular arithmetic operations are considered very fast. Thus, we just count exponent operation and pairing operation when analyzing the performance of the proposed protocol. We compare the computation overheads between batch and individual auditing. Here, the task is a batch of auditing for K users, referring to C cloud servers; and t is the number of challenged data blocks. The result is shown in Table 2, where [E] denotes the computation cost of an exponent operation in G1, G2 or GT, and [B] denotes the computation cost of a bilinear map $e(\cdot,\cdot)$.

Table 2. Comparison on Computation Overheads between Batch and Individual Auditing

Protocol	*KeyGen*	*TagGen*	*GenProof*(*P*)	Verification(TPA)
Individual auditing	*K*[*E*]	*K*(*n*+1)(*s*+1) [*E*]	*KCs*[*B*]+*K*(*t*+*s*)[*E*]	4*K*[*B*]+*K*(*t*+*s*)[*E*]
Batch auditing	*K*[*E*]	*K*(*n*+1)(*s*+1) [*E*]	*KCs*[*B*]+*K*(*t*+*s*)[*E*]	(2*K*+2)[*B*]+*K*(*t*+*s*)[*E*]

As shown in Eq. (10) and Eq. (11), batch auditing not only allows TPA to perform the multiple auditing tasks simultaneously, but also greatly reduces the computation cost on the TPA side. By aggregating *K* verification equations into one, the proposed protocol reduces the number of pairing operations from 4*K* (as required in the individual auditing) to 2*K*+2, which saves a considerable amount of auditing time for TPA. Actually, the communication overhead between the cloud servers and TPA is also greatly reduced. Besides our protocol greatly reduce the computing overhead of the TPA by utilizing cloud servers to compute intermediate values of the verification for the TPA.

Communication Cost: In this section, we give the comparison of communication cost between our protocol and two existing works: the CPDP proposed by Zhu, *et al.*, [12] and the batch audit protocol proposed by Yang and Jia [15]. The communication cost of the setup phase is almost the same in these protocols. The communication cost between the auditor and the server in audit phase, which consists of the challenge and the proof, is described in Table 3.

Table 3. The Communication Cost Comparison of Batch Auditing for k users and c Cloud Servers

Protocol	*Challenge*	*Proof*
CPDP protocol [12]	$O(KCt)$	$O(KCs)$
Batch Auditing protocol [15]	$O(KCt)$	$O(C)$
Our Protocol	$O(KCt)$	$O(C)$

The three protocols have the same communication cost in the challenge phase. During the proof phase, like the literature [15], the communication cost of the proof is only linear to C in our protocol. But the literature [15] didn't consider stateless verification and transparent verification. On the other side, in Zhu's CPDP, the communication cost of the proof is not only linear to C and K, but also linear to s. That is because Zhu's CPDP does not support the batch auditing for multiple users, and it uses the mask technique to protect the data privacy, which requires sending both the masked proof and the encrypted mask to auditor.

5. Conclusions

This paper develops an efficient auditing mechanism, which supports batch auditing for multiple data files in multi-cloud environment. By utilizing the bilinear map, the proposed protocol achieves full stateless and transparent verification. By constructing a sequence-enforced Merkle Hash Tree, the proposed protocol can support dynamic operation of data and resist the replace attack. In addition, our protocol protects the position information of the data blocks by generating fake data blocks to confuse the organizer. By computing intermediate values of the verification on cloud servers, our method can greatly reduce the computing overhead of the auditor. The performance analysis proves the good efficiency of the proposed protocol. It is worth noting that, the organizer in the proposed protocol can be easily removed by transmitting the organizer's tasks to TPA. This may increase the burden of TPA. However, a model without the organizer may correspond to reality better.

Acknowledgements

This work is supported by the 12th five year programming of scientific research in education of Hunan Province (XJKOIICTM15), and Scientific Research Fund of Hunan Provincial Education Department (11C0385).

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable data possession at untrusted stores", Proceedings of the 14th ACM conference on Computer and communications security, **(2007)**, pp. 598-609.

[2] G. Ateniese, R. Di Pietro, L. V. Mancini and G. Tsudik, "Scalable and efficient provable data possession", Proceedings of the 4th international conference on Security and privacy in communication netowrks, NY, USA, **(2008)**, pp. 9.

[3] C. Erway, A. Küpçü, C. Papamanthou and R. Tamassia, "Dynamic provable data possession", Proceedings of the 16th ACM conference on Computer and communications security, **(2009)**, pp. 213-222.
[4] Q. Wang, C. Wang, K. Ren, W. Lou and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing", Parallel and Distributed Systems, IEEE Transactions on, vol. 22, no. 5, **(2011)**, pp. 847-859.
[5] D. Boneh, B. Lynn and H. Shacham, "Short signatures from the Weil pairing", In Advances in Cryptology ASIACRYPT 2001, Springer, **(2001)**, pp. 514-532.
[6] C. Wang, S. Chow, Q. Wang, K. Ren and W. Lou, "Privacy-preserving public auditing for secure cloud storage", Computers, IEEE Transactions on, vol. 62, no. 2, **(2013)**, pp. 362-375.
[7] A. Juels and B. S. Kaliski Jr, "PORs: Proofs of retrievability for large files", Proceedings of the 14th ACM conference on Computer and communications security, **(2007)**, pp. 584-597.
[8] H. Kai, H. Chuanhe, W. Jinhai, Z. Hao, C. Xi, L. Yilong, Z. Lianzhen and W. Bin, "An Efficient Public Batch Auditing Protocol for Data Security in Multi-cloud Storage", 8th ChinaGrid Annual Conference, **(2013)**, pp. 51-56.
[9] H. Wang and Y. Zhang, "On the Knowledge Soundness of a Cooperative Provable Data Possession Scheme in Multicloud Storage", Parallel and Distributed Systems, IEEE Transactions on, vol. 25, no. 1, **(2013)**, pp. 264-267.
[10] M. A. AlZain, E. Pardede, B. Soh and J. A. Thom, "Cloud computing security: from single to multi-clouds", 45th Hawaii International Conference on System Science, **(2012)**, pp. 5490-5499.
[11] Y. Zhu, H. Hu, G.-J. Ahn, Y. Han and S. Chen, "Collaborative integrity verification in hybrid clouds", 7th International Conference on Collaborative Computing: Networking, Applications and Worksharing, **(2011)**, pp. 191-200.
[12] Y. Zhu, H. Hu, G. Ahn and M. Yu, "Cooperative provable data possession for integrity verification in multi-cloud storage", IEEE Transactions on Parallel and Distributed Systems, vol. 23, no. 12, **(2012)**, pp. 2231-2244.
[13] X. Sun, L. Chen, Z. Xia and Y. Zhu, "Cooperative Provable Data Possession with Stateless Verifcation in Multicloud Storage", Journal of Computational Information Systems (JCIS), vol. 10, no. 8, **(2014)**, pp. 1-9.
[14] M. Etemad and A. Küpçü, "Transparent, Distributed, and Replicated Dynamic Provable Data Possession", Applied Cryptography and Network Security, **(2013)**, pp. 1-18.
[15] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing", Parallel and Distributed Systems, IEEE Transactions on, vol. 24, no. 9, **(2013)**, pp. 1717-1726.
[16] R. C. Merkle, "Protocols for Public Key Cryptosystems", IEEE Symposium on Security and privacy, **(1980)**, pp. 122-134.
[17] A. L. Ferrara, M. Green, S. Hohenberger and M. Ø. Pedersen, "Practical short signature batch verification", In Topics in Cryptology–CT-RSA 2009, Springer Berlin Heidelberg, **(2009)**, pp. 309-324.

Authors

Xin Liu, he received his BS at Artistic Design Department of Jilin Teacher's College of Engineering and Technology, China, in 2001, MS at Artistic Design Department of Jilin Art College, China, in 2004. He works as a lecturer at College of Architecture and Artistic Design Department in Hunan Institute of Technology. His research interests include information security, image processing, and animation production.

Yujia Jiang, she received his BS at Artistic Design Department of Jilin Teacher's College of Engineering and Technology, China, in 2001, MS at Artistic Design Department of Jilin Art College, China, in 2005. She works as a lecturer at College of Architecture and Artistic Design Department in Hunan Institute of Technology. Hers research interests include information security, image processing, and artistic design.